CLAIMS

1. A process for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising the steps of:

providing a streaming file system on said client;

wherein said streaming file system appears to said client to contain the installed application program;

wherein said streaming file system receives all requests from local processes for application program code or data that are part of the application;

providing a persistent cache on said client;

wherein said streaming file system satisfies requests for application program code or data by retrieving it from said persistent cache stored in a native file system or by retrieving it directly from said server; and

wherein application program code or data retrieved from said server is placed in said persistent cache for reuse.

2. The process of claim 1, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

3. The process of claim 1, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

4. The process of claim 1, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

5. The process of claim 1, further comprising the step of:

providing a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

6. The process of claim 1, further comprising the step of:

wherein said streaming file system is a copy-on-write file system that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

providing a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write file system references said cache index to determine if a page is clean or dirty.

7. The process of claim 1, further comprising the step of:

marking specific files in said persistent cache as not modifiable;

wherein said streaming file system does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

8. The process of claim 1, further comprising the step of:

maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

9. The process of claim 1, further comprising the step of:

assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

10. The process of claim 9, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said streaming file system to search for files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

11. The process of claim 9, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said streaming file system contacts said server when an application program is started in order to receive any application upgrades.

12. The process of claim 1, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

13. A process for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising the steps of:

providing a kernel-mode streaming file system driver on said client;

providing a user-mode client on said client;

wherein said streaming file system receives all requests from local processes for application program code or data that are part of the application;

providing a persistent cache on said client;

wherein requests made to said streaming file system are directed to said user-mode client or retrieved from said persistent cache;

wherein said user-mode client handles the application program code and data streams from said server and sends the results back to said streaming file system driver; and

wherein application program code or data retrieved from said server is placed in said persistent cache for reuse.

14. The process of claim 13, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

15. The process of claim 13, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

16. The process of claim 13, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

17. The process of claim 13, further comprising the step of:

providing a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

18. The process of claim 13, further comprising the step of:

wherein said streaming file system is a copy-on-write file system that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

providing a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write file system references said cache index to determine if a page is clean or dirty.

19. The process of claim 13, further comprising the step of:

marking specific files in said persistent cache as not modifiable;

wherein said streaming file system does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

20. The process of claim 13, further comprising the step of:

maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

21. The process of claim 13, further comprising the step of:

assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

22. The process of claim 21, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said streaming file system to search for files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

23. The process of claim 21, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said streaming file system contacts said server when an application program is started in order to receive any application upgrades.

24. The process of claim 13, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

25. A process for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising the steps of:

providing a streaming block driver on said client;

wherein said block driver provides the abstraction of a physical disk to a native file system already installed on the client operating system;

providing a persistent cache on said client;

wherein said block driver receives requests for physical block reads and writes from local processes which it satisfies out of said persistent cache on a standard file system that is backed by a physical disk drive; and

wherein requests that cannot be satisfied by said persistent cache are sent to said server.

26. The process of claim 25, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or

duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

27. The process of claim 25, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

28. The process of claim 25, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

29. The process of claim 25, further comprising the step of:

providing a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

30. The process of claim 25, further comprising the step of:

wherein said block driver is a copy-on-write block driver that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

providing a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write block driver references said cache index to determine if a page is clean or dirty.

31. The process of claim 25, further comprising the step of:

marking specific files in said persistent cache as not modifiable;

wherein said block driver does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

32. The process of claim 25, further comprising the step of:

maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

33. The process of claim 25, further comprising the step of:

assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

34. The process of claim 33, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said block driver to search for files in the application program; wherein files that do not change can be reused from said

persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

35. The process of claim 33, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said block driver contacts said server when an application program is started in order to receive any application upgrades.

36. The process of claim 25, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

37. A process for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising the steps of:

providing a disk driver on said client;

providing a user mode client on said client;

wherein said disk driver sends all file requests that it receives to said user-mode client;

providing a persistent cache on said client; and

wherein said user-mode client attempts to satisfy said file requests from said program cache or by making requests from said server.

38. The process of claim 37, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

39. The process of claim 37, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

40. The process of claim 37, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

41. The process of claim 37, further comprising the step of:

providing a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

42. The process of claim 37, further comprising the step of:

wherein said user-mode client is a copy-on-write user-mode client that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

providing a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write user-mode client references said cache index to determine if a page is clean or dirty.

43. The process of claim 37, further comprising the step of:

marking specific files in said persistent cache as not modifiable;

wherein said user-mode client does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

44. The process of claim 37, further comprising the step of:

maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

45. The process of claim 37, further comprising the step of:

assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

46. The process of claim 45, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said user-mode client to search for files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

47. The process of claim 45, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said user-mode client contacts said server when an application program is started in order to receive any application upgrades.

48. The process of claim 37, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

49. An apparatus for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising:

a streaming file system on said client;

wherein said streaming file system appears to said client to contain the installed application program;

wherein said streaming file system receives all requests from local processes for application program code or data that are part of the application;

a persistent cache on said client;

wherein said streaming file system satisfies requests for application program code or data by retrieving it from said persistent cache stored in a native file system or by retrieving it directly from said server; and

wherein application program code or data retrieved from said server is placed in said persistent cache for reuse.

50. The apparatus of claim 49, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

51. The apparatus of claim 49, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

52. The apparatus of claim 49, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

53. The apparatus of claim 49, further comprising:

a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

54. The apparatus of claim 49, further comprising:

wherein said streaming file system is a copy-on-write file system that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write file system references said cache index to determine if a page is clean or dirty.

55. The apparatus of claim 49, further comprising:

a module for marking specific files in said persistent cache as not modifiable;

wherein said streaming file system does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

56. The apparatus of claim 49, further comprising:

a module for maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

57. The apparatus of claim 49, further comprising:

a module for assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

58. The apparatus of claim 57, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said streaming file system to search for files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

59. The apparatus of claim 57, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said streaming file system contacts said server when an application program is started in order to receive any application upgrades.

60. The apparatus of claim 49, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

61. An apparatus for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising:

a kernel-mode streaming file system driver on said client;

a user-mode client on said client;

wherein said streaming file system receives all requests from local processes for application program code or data that are part of the application;

a persistent cache on said client;

wherein requests made to said streaming file system are directed to said user-mode client or retrieved from said persistent cache;

wherein said user-mode client handles the application program code and data streams from said server and sends the results back to said streaming file system driver; and

wherein application program code or data retrieved from said server is placed in said persistent cache for reuse.

62. The apparatus of claim 61, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

63. The apparatus of claim 61, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

64. The apparatus of claim 61, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

65. The apparatus of claim 61, further comprising:

a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

66. The apparatus of claim 61, further comprising:

wherein said streaming file system is a copy-on-write file system that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write file system references said cache index to determine if a page is clean or dirty.

67. The apparatus of claim 61, further comprising:

a module for marking specific files in said persistent cache as not modifiable;

wherein said streaming file system does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

68. The apparatus of claim 61, further comprising:

a module for maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

69. The apparatus of claim 61, further comprising:

a module for assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

70. The apparatus of claim 69, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said streaming file system to search for files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

71. The apparatus of claim 69, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said streaming file system contacts said server when an application program is started in order to receive any application upgrades.

72. The apparatus of claim 61, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

73. An apparatus for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising:

a streaming block driver on said client;

wherein said block driver provides the abstraction of a physical disk to a native file system already installed on the client operating system;

a persistent cache on said client;

wherein said block driver receives requests for physical block reads and writes from local processes which it satisfies out of said persistent cache on a standard file system that is backed by a physical disk drive; and

wherein requests that cannot be satisfied by said persistent cache are sent to said server.

74. The apparatus of claim 73, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or

duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

75. The apparatus of claim 73, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

76. The apparatus of claim 73, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively retums to said client additional blocks that said client did not request but is likely to need soon.

77. The apparatus of claim 73, further comprising:

a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

78. The apparatus of claim 73, further comprising:

wherein said block driver is a copy-on-write block driver that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write block driver references said cache index to determine if a page is clean or dirty.

79. The apparatus of claim 73, further comprising:

a module for marking specific files in said persistent cache as not modifiable;

wherein said block driver does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

80. The apparatus of claim 73, further comprising:

a module for maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

81. The apparatus of claim 73, further comprising:

a module for assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

82. The apparatus of claim 81, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said block driver to search for

files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

83. The apparatus of claim 81, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said block driver contacts said server when an application program is started in order to receive any application upgrades.

84. The apparatus of claim 73, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.

85. An apparatus for client-side retrieval, storage, and execution of application programs and other related data streamed from a server across a computer network to a client system in a computer environment, comprising:

a disk driver on said client;

a user mode client on said client;

wherein said disk driver sends all file requests that it receives to said user-mode client;

a persistent cache on said client; and

wherein said user-mode client attempts to satisfy said file requests from said program cache or by making requests from said server.

86. The apparatus of claim 85, wherein said persistent cache is encrypted with a key not permanently stored on said client to prevent unauthorized use or duplication of application code or data; and wherein said key is sent to said client upon application startup from said server and said key is not stored in the application program's persistent storage area in said persistent cache.

87. The apparatus of claim 85, wherein said client initiates the prefetching of application program code and data from said server; and wherein said client inspects program code or data file requests and consults the contents of said persistent cache as well as historic information about application program fetching patterns and uses this information to request additional blocks of application program code and data from said server that said client expects will be needed soon.

88. The apparatus of claim 85, wherein said server initiates the prefetching of application program code and data for said client; and wherein said server examines the patterns of requests made by said client and selectively returns to said client additional blocks that said client did not request but is likely to need soon.

89. The apparatus of claim 85, further comprising:

a client-to-client communication mechanism that allows local application customization to migrate from one client machine to another without involving server communication;

wherein when a user wishes to run an application on a second machine, but wishes to retain customizations made previously on the first, said client-to-client mechanism contacts the first machine to retrieve customized files and other customization data; and

wherein unmodified files are retrieved from said server.

90. The apparatus of claim 85, further comprising:

wherein said user-mode client is a copy-on-write user-mode client that allows applications to write configuration or initialization files where they want to without rewriting the application, and without disturbing the local customization of other clients;

wherein each block of data in said persistent cache is marked as clean or dirty;

wherein pages marked as dirty have been customized by the application program and cannot be removed from the cache without losing client customization;

wherein pages marked as clean may be purged from the cache because they can be retrieved again from said server;

a cache index;

wherein said cache index indicates which pages in said persistent cache are clean and dirty; and

wherein said copy-on-write user-mode client references said cache index to determine if a page is clean or dirty.

91. The apparatus of claim 85, further comprising:

a module for marking specific files in said persistent cache as not modifiable;

wherein said user-mode client does not allow any data to be written to said specific files that are marked as not modifiable; and

wherein attempts by any processes to mark any of said specific files as modifiable will not succeed.

92. The apparatus of claim 85, further comprising:

a module for maintaining checksums of application code and data in said persistent cache;

wherein when a block of code or data is requested by a local process said streaming file system computes the checksum of the data block before it is returned to the local process; and

wherein if a computed checksum does not match the checksum stored in said persistent cache the cache entry is invalidated and a fresh copy of the page is retrieved from said server.

93. The apparatus of claim 85, further comprising:

a module for assigning each file in an application program a unique identifier;

wherein files that are changed or added in an application upgrade are given new identifiers never before used for that application program;

wherein files that are unchanged retain the same number; and

wherein directories whose contents change are also considered changes. If any file changes, this will cause its parent to change, all the way up to the root directory.

94. The apparatus of claim 93, wherein when an application upgrade occurs said client is given a new root directory for the application program by said server; wherein said new root directory is used by said user-mode client to search for files in the application program; wherein files that do not change can be reused from said persistent cache without downloading them again from said server; and wherein files with new identifiers are retrieved from said server.

95. The apparatus of claim 93, wherein said application upgrades can be marked as mandatory by said server causing the new root directory for the application program to be used immediately; and wherein said user-mode client contacts said server when an application program is started in order to receive any application upgrades.

96. The apparatus of claim 85, wherein said server broadcasts an application program's code and data and any client that is interested in that particular application program stores the broadcasted code and data for later use.